15-814 Homework 4

October 23, 2017

1 Abstraction

1.1 Abstraction Theorem

Task 1 *Show that there are no expressions e of type $\forall\alpha.\forall\beta.\alpha \to \beta$.*

Solution: Suppose that there is an expression e of type $\forall\alpha.\forall\beta.\alpha \to \beta$. By the abstraction theorem, we have $e \sim e : \forall\alpha.\forall\beta.\alpha \to \beta$.

Consider the relation $R : 2 \leftrightarrow 2 = \{(e_1, e_2) \mid e_1 \cong e_2\}$, where $\cong$ is contextual equivalence[1]. In particular, R is non-empty, and we have $(\texttt{tt}, \texttt{tt}) \in R$. Unfolding the definition of $\sim$ (and picking the candidate), we get:

$$e[2] \sim e[2] : \forall\beta.(2, 2, R) \to \beta$$

Next, consider the empty relation $Q : 2 \leftrightarrow 2 = \emptyset$. Expanding the definition of $\sim$ again, we get:

$$e[2][2] \sim e[2][2] : (2, 2, R) \to (2, 2, Q)$$

From the observation above, we also have:

$$\texttt{tt} \sim \texttt{tt} : (2, 2, R)$$

Therefore by definition of $\sim$,

$$e[2][2]\texttt{tt} \sim e[2][2]\texttt{tt} : (2, 2, Q)$$

This yields a contradiction, because Q is the empty relation but we have:

$$(e[2][2]\texttt{tt}, e[2][2]\texttt{tt}) \in Q$$

Task 2 *Show that for any expression e of type $\forall\alpha.\alpha \to \alpha \to \alpha$ and any $e_1 : \tau$ and $e_2 : \tau$, either $e[\tau](e_1)(e_2) \sim e_1 : \tau$ or $e[\tau](e_1)(e_2) \sim e_2 : \tau$.*

Solution: By the abstraction theorem, we have $e \sim e : \forall\alpha.\alpha \to \alpha \to \alpha$. Consider the relation $R : \tau \leftrightarrow \tau = \{(x, y) \mid x \sim e_1 \lor x \sim e_2\}$, we write $R_\tau = (\tau, \tau, R)$ for the corresponding candidate. Expanding the definition of $\sim$, we get:

$$e[\tau] \sim e[\tau] : R_\tau \to R_\tau \to R_\tau$$

From the abstraction theorem, we also have $e_1 \sim e_1 : \tau$ and $e_2 \sim e_2 : \tau$. Therefore, by definition of $\sim$, we have $e_1 \sim e_1 : R_\tau$ and $e_2 \sim e_2 : R_\tau$. This yields:

$$e[\tau](e_1)(e_2) \sim e[\tau](e_1)(e_2) : R_\tau$$

Expanding R_τ, we get $e[\tau](e_1)(e_2) \sim e_1 \lor e[\tau](e_1)(e_2) \sim e_2$.

[1]This is just to make R admissible, see PFPL 48.

1.2 Data Abstraction

Task 3 *State the abstract type of the signature queue.*

Solution:
Curried:

$$\forall\beta.(\forall\alpha.\alpha \to (\texttt{int} \times \alpha \to \alpha) \to (\alpha \to (\texttt{unit} + \texttt{int} \times \alpha)) \to \beta) \to \beta$$

Uncurried:

$$\forall\beta.(\forall\alpha.\alpha \times (\texttt{int} \times \alpha \to \alpha) \times (\alpha \to (\texttt{unit} + \texttt{int} \times \alpha)) \to \beta) \to \beta$$

Alternatively, with existential types:

$$\exists\alpha.\alpha \times (\texttt{int} \times \alpha \to \alpha) \times (\alpha \to (\texttt{unit} + \texttt{int} \times \alpha))$$

Task 4 *State an admissible relation* $R : \texttt{int list} \leftrightarrow \texttt{int list} \times \texttt{int list}$ *which is closed under all operations of the signature.*

Solution: Intuitively, LQ keeps a list for its queue implementation, and in LLQ, we can recover this list by reversing the second queue and appending it to the first queue at any time during the queue's operation.

$$R : \texttt{int list} \leftrightarrow \texttt{int list} \times \texttt{int list} = \{(l, (l_1, l_2)) \mid l \cong \texttt{append}\ l_1\ (\texttt{rev}\ l_2)\}$$

Note that $\cong$ is (1) closed under evaluation, (2) an equivalence relation, and (3) a congruence. Define the candidate $Q = (\texttt{int list}, \texttt{int list} \times \texttt{int list}, R)$ for the next parts of this section. The operations of LQ are written with subscript 1 while those of LLQ are written with subscript 2. I ignore the typing derivations for $\sim$, but all of these parts are also obviously well-typed.

Task 5 *Prove that the relation R you defined above is closed under emp.*

Solution: We need to show $\texttt{emp}_1 \sim \texttt{emp}_2 : Q$, i.e. that $[\,]\ R\ ([\,], [\,])$. We have:

$$\texttt{append}\ [\,]\ (\texttt{rev}\ [\,]) \cong \texttt{append}\ [\,]\ ([\,]) \cong [\,]$$

.

Task 6 *Prove that the relation R you defined above is closed under ins.*

Solution: We need to show $\texttt{ins}_1 \sim \texttt{ins}_2 : (\texttt{int} \times Q \to Q)$. Consider inputs $(n, l) \sim (n', (l_1, l_2)) : \texttt{int} \times Q$. By definition of the ins functions, and $\sim$ for function types, we need to show:

$$\texttt{ins}_1\ (n, l)\ R\ \texttt{ins}_2\ (n', (l1, l2))$$

$$\texttt{append}\ l\ [n] \cong \texttt{append}\ l_1\ (\texttt{rev}(n' :: l_2))$$

Note that the definition of $\sim$ for products implies $n \sim n'$, and so we have $n \cong n'$. Therefore, by congruence and transitivity, we only need to show:

$$\texttt{append}\ l\ [n] \cong \texttt{append}\ l_1\ (\texttt{rev}(n :: l_2))$$

Observe that:

$$\texttt{rev}(n :: l_2) \cong \texttt{append}\ (\texttt{rev}\ l_2)\ [n]$$

and so by congruence and associativity of append:

$$\texttt{append}\ l_1\ (\texttt{rev}(n :: l_2)) \cong \texttt{append}\ (\texttt{append}\ l_1\ (\texttt{rev}\ l_2))\ [n] \cong \texttt{append}\ l\ [n]$$

The latter equivalence holds by congruence and the assumption that the input queues are related at Q, i.e. :

$$\texttt{append}\ l_1\ (\texttt{rev}\ l_2) \cong l$$

Task 7 *Prove that the relation R you defined above is closed under rem.*

Solution: We need to show $\texttt{rem}_1 \sim \texttt{rem}_2 : (Q \to (\texttt{unit} + \texttt{int} \times Q))$. Consider inputs $l \sim (l_1, l_2) : Q$. Unfolding definitions as before, we need to show:

$$\texttt{rem}_1 l \cong \texttt{rem}_2(l_1, l_2)$$

We follow the case analysis on l in $\texttt{rem}_1$.

Case [ ]: Since l and (l_1, l_2) are related at Q, we have:

$$\texttt{append}\ l_1\ (\texttt{rev}\ l_2) \cong [\,]$$

Thus, (l_1, l_2) must both evaluate to [ ]. Therefore, both $\texttt{rem}_1$ and $\texttt{rem}_2$ evaluate to NONE, which is contextually equivalent to itself.

Case $(x :: xs)$: $\texttt{rem}_1$ evaluates to SOME(x, xs). By assumption that the inputs are related at Q, we also have:

$$\texttt{append}\ l_1\ (\texttt{rev}\ l_2) \cong (x :: xs)$$

Now we may case analyze on l_1:

Subcase $l_1 \cong [\,]$. We have

$$\texttt{append}\ l_1\ (\texttt{rev}\ l_2) \cong \texttt{rev}\ l_2 \cong x :: xs$$

Therefore, we have

$$\texttt{rem}_2(l_1, l_2) \cong \texttt{rem}_2(\texttt{rev}\ l_2, [\,]) \cong \texttt{rem}_2(x :: xs, [\,]) \cong \text{SOME}(x, (xs, [\,]))$$

Now we have $xs \cong \texttt{append}\ xs\ (\texttt{rev}\ [\,])$, and therefore, SOME$(x, xs) \cong$ SOME$(x, (xs, [\,]))$ so we are done.

Task 8 *Briefly compare the two implementations of queue in terms of time efficiency. Are there situations where you would prefer one over the other?*

Solution: The second queue implementation has amortized constant time complexity but is potentially slow on some calls when it needs to reverse the second list. The first implementation is linear in the size of the list for each insertion, but is constant time for removal. We would use the second implementation for most purposes, but may resort to the first if we want to have more predictable behavior.

2 Free Theorems

Task 9 *Consider a function* $\texttt{h} : \forall\alpha.\forall\beta.\alpha\ \texttt{list} \times \beta\ \texttt{list} \to (\alpha \times \beta)\ \texttt{list}$ *(with the type of the standard zip function). Consider functions* $\texttt{a} : \tau_A \to \tau'_A$ *and* $\texttt{b} : \tau_B \to \tau'_B$*. Let* $\texttt{a}^*$ *(similarly for* $\texttt{b}^*$*) denote the map of* $\texttt{a}$ *over a list of type* τ_A*, i.e.* $\texttt{a}^*$ *takes a list* $\texttt{l}$ *as argument and applies* $\texttt{a}$ *to every element of* $\texttt{l}$*. Also, let* $\texttt{a} \times \texttt{b} : \tau_A \times \tau_B \to \tau'_A \times \tau'_B$ *be the function where* $(\texttt{a} \times \texttt{b})\langle \texttt{e}_1, \texttt{e}_2 \rangle = \langle \texttt{a}\ \texttt{e}_1, \texttt{b}\ \texttt{e}_2 \rangle$*. Prove that for all* $e : \tau_A\ \texttt{list} \times \tau_B\ \texttt{list}$*:*

$$(\texttt{a} \times \texttt{b})^*(\texttt{h}[\tau_A][\tau_B]\ e) \sim \texttt{h}[\tau'_A][\tau'_B]((\texttt{a}^* \times \texttt{b}^*)\ e) : (\tau'_A \times \tau'_B)\ \texttt{list}$$

Solution: Let $P = (\tau_A, \tau'_A, \texttt{a})$, $Q = (\tau_B, \tau'_B, \texttt{b})$, where we view the functions $\texttt{a}, \texttt{b}$ as relations. By the abstraction theorem,

$$\texttt{h} \sim \texttt{h} : \forall\alpha.\forall\beta.\alpha\ \texttt{list} \times \beta\ \texttt{list} \to (\alpha \times \beta)\ \texttt{list}$$

Expanding the definition of $\sim$, we have:

$$\texttt{h}[\tau_A][\tau_B] \sim \texttt{h}[\tau'_A][\tau'_B] : P\ \texttt{list} \times Q\ \texttt{list} \to (P \times Q)\ \texttt{list}$$

Let $e : \tau_A\ \texttt{list} \times \tau_B\ \texttt{list}$, then we know[2] that $\pi_1 e \sim \texttt{a}^*(\pi_1 e) : P\ \texttt{list}$ and $\pi_2 e \sim \texttt{b}^*(\pi_2 e) : Q\ \texttt{list}$.

[2]As we saw in class, mapping a function $\texttt{a}$ over a list l gives us $l \sim \texttt{a}^* l : P$.

Therefore, since $(\mathtt{a}^* \times \mathtt{b}^*)$ applies to $\mathtt{a}^*, \mathtt{b}^*$ to the left and right projections of e respectively, we also have:

$$e \sim ((\mathtt{a}^* \times \mathtt{b}^*)\ e) : P\ \mathtt{list} \times Q\ \mathtt{list}$$

Expanding the definition of $\sim$ again,

$$\mathtt{h}[\tau_A][\tau_B]e \sim \mathtt{h}[\tau_A'][\tau_B']((\mathtt{a}^* \times \mathtt{b}^*)\ e) : (P \times Q)\ \mathtt{list}$$

Now, by expanding the definition of $(P \times Q)$ `list`, we have that the two lists are element-wise related at $P \times Q$. In other words, for each element (e_1, e_2) of the left list, its corresponding element in the right list is $(\mathtt{a}\ e_1, \mathtt{b}\ e_2)$. Therefore, by mapping $\mathtt{a} \times \mathtt{b}$ over the left list, we have[3]:

$$(\mathtt{a} \times \mathtt{b})^*(\mathtt{h}[\tau_A][\tau_B]\ e) \sim \mathtt{h}[\tau_A'][\tau_B']((\mathtt{a}^* \times \mathtt{b}^*)\ e) : (\tau_A' \times \tau_B')\ \mathtt{list}$$

[3]One could also argue this more precisely by expanding out the definition of $\sim$ for lists.